Pengantar Pemrograman C#

[Lecture Note Pertemuan ke - 04]
(Percabangan & Perulangan)

Ng Poi Wong, 2019, Sesi 2 : Percabangan & Perulangan, Lecture Notes, Pengantar Pemrograman C# (IF0054), STMIK Mikroskil Medan, Dikirimkan 02 September 2019.

Capaian MK : Mahasiswa mampu mengimplementasikan konsep percabangan dan perulangan

DAFTAR ISI PENJELASAN SLIDE

PENJELASAN DARI SLIDE ke-3 & 4

[Slide 3]

- Merupakan daftar operator logika pada Visual C#.
- "`cek = a == b;`" → Jika nilai variabel **a** sama dengan nilai variabel **b**, maka variabel **cek** akan bernilai **True**, dan sebaliknya akan bernilai **False**.
- "`cek = a != b;`" → Jika nilai variabel **a** tidak sama dengan nilai variabel **b**, maka variabel **cek** akan bernilai **True**, dan sebaliknya akan bernilai **False**.
- "`cek = a < b;`" → Jika nilai variabel **a** lebih kecil dari nilai variabel **b**, maka variabel **cek** akan bernilai **True**, dan sebaliknya akan bernilai **False**.
- "`cek = a > b;`" → Jika nilai variabel **a** lebih besar dari nilai variabel **b**, maka variabel **cek** akan bernilai **True**, dan sebaliknya akan bernilai **False**.
- "`cek = a <= b;`" → Jika nilai variabel **a** lebih kecil atau sama dengan nilai variabel **b**, maka variabel **cek** akan bernilai **True**, dan sebaliknya akan bernilai **False**.
- "`cek = a >= b;`" → Jika nilai variabel **a** lebih besar atau sama dengan nilai variabel **b**, maka variabel **cek** akan bernilai **True**, dan sebaliknya akan bernilai **False**.

[Slide 4]

Dari contoh kode program tersebut, dapat dijelaskan :

- "`var1 = var2 != var3;`" → Nilai variabel **var2** (bernilai 10) tidak sama dengan nilai **var3** (bernilai 3), maka variabel **var1** akan bernilai **True**.
- "`var1 = var2 < var3;`" → Nilai variabel **var2** (bernilai 10) TIDAK lebih kecil dari nilai **var3** (bernilai 3), maka variabel **var1** akan bernilai **False**.
- "`var1 = var2 > var3;`" → Nilai variabel **var2** (bernilai 10) lebih besar dari nilai **var3** (bernilai 3), maka variabel **var1** akan bernilai **True**.

PENJELASAN DARI SLIDE ke-5 s/d 8

[Slide 5 & 6]

- Merupakan daftar operator logika lanjutan pada Visual C#.
- "`cek = !a;`" → Nilai variabel **a** akan dilakukan logika **NOT** dan hasilnya disimpan ke dalam variabel **cek**.
- "`cek = a & b;`" → Nilai variabel **a** dan **b** akan dilakukan logika **AND** dan hasilnya disimpan ke dalam variabel **cek**.
- "`cek = a | b;`" → Nilai variabel **a** dan **b** akan dilakukan logika **OR** dan hasilnya disimpan ke dalam variabel **cek**.
- "`cek = a ^ b;`" → Nilai variabel **a** dan **b** akan dilakukan logika **XOR** dan hasilnya disimpan ke dalam variabel **cek**.

[Slide 7 & 8]

Dari contoh kode program tersebut, dapat dijelaskan :

- "`var2 = !var3;`"
  Nilai variabel **var3** (bernilai **True**) akan dilakukan logika **NOT**, dan hasilnya disimpan ke variabel **var2** dengan nilai **False**.
- "`var1 = var2 & var3;`"
  Nilai variabel **var2** (sudah berubah menjadi nilai **False**) akan dilakukan logika **AND** dengan nilai **var3** (bernilai **True**), dan hasilnya disimpan ke variabel **var1** dengan nilai **False**.
- "`var2 |= var3;`"
  Identik dengan "`var2 = var2 | var3;`", dimana nilai variabel **var2** (bernilai **False**) akan dilakukan logika **OR** dengan nilai **var3** (bernilai **True**), dan hasilnya disimpan kembali ke variabel **var2** dengan nilai **True**.
- "`var2 ^= var3;`"
  Identik dengan "`var2 = var2 ^ var3;`", dimana nilai variabel **var2** (sudah berubah menjadi nilai **True**) akan dilakukan logika **OR** dengan nilai **var3** (bernilai **True**), dan hasilnya disimpan kembali ke variabel **var2** dengan nilai **False**.

PENJELASAN DARI SLIDE ke-9 & 10

[Slide 9]

- Operator logika **&** dengan **&&** merupakan logika **AND**.
- Operator logika **|** dengan **||** merupakan logika **OR**.
- Perbedaan antara **&** dengan **&&** adalah :
  - Jika terdapat ekspresi1 **&&** ekspresi2, maka jika hasil evaluasi ekspresi1 bernilai **False**, maka ekspresi2 tidak akan dievaluasi lagi dan langsung dikembalikan nilai **False**, tetapi sebaliknya jika hasil evaluasi ekspresi1 bernilai **True**, maka akan dilanjutkan lagi evaluasi ekspresi2 untuk mendapatkan hasil logika **AND**.
  - Jika terdapat ekspresi1 **&&** ekspresi2, serta ekspresi1 dan ekspresi2 berupa variabel, maka variabel tersebut wajib berupa variabel bertipe **Boolean (bool)**.
  - Jika terdapat ekspresi1 **&** ekspresi2, maka jika hasil evaluasi ekspresi1 bernilai **False**, maka ekspresi2 akan tetap dievaluasi untuk mendapatkan hasil logika **AND**.
  - Jika terdapat ekspresi1 **&** ekspresi2, serta ekspresi1 dan ekspresi2 berupa variabel, maka variabel tersebut dapat berupa variabel bertipe **Boolean (bool)** maupun **bilangan bulat**. Jika berupa bilangan bulat, maka logika **AND** dilakukan terhadap bilangan biner dari bilangan bulat tersebut.
- Perbedaan antara **|** dengan **||** adalah :
  - Jika terdapat ekspresi1 **||** ekspresi2, maka jika hasil evaluasi ekspresi1 bernilai **True**, maka ekspresi2 tidak akan dievaluasi lagi dan langsung dikembalikan nilai **True**, tetapi sebaliknya jika hasil evaluasi ekspresi1 bernilai **False**, maka akan dilanjutkan lagi evaluasi ekspresi2 untuk mendapatkan hasil logika **AND**.
  - Jika terdapat ekspresi1 **||** ekspresi2, serta ekspresi1 dan ekspresi2 berupa variabel, maka variabel tersebut wajib berupa variabel bertipe **Boolean (bool)**.
  - Jika terdapat ekspresi1 **|** ekspresi2, maka jika hasil evaluasi ekspresi1 bernilai **True**, maka ekspresi2 akan tetap dievaluasi untuk mendapatkan hasil logika **AND**.
  - Jika terdapat ekspresi1 **|** ekspresi2, serta ekspresi1 dan ekspresi2 berupa variabel, maka variabel tersebut dapat berupa variabel bertipe **Boolean (bool)** maupun **bilangan bulat**. Jika berupa bilangan bulat, maka logika **OR** dilakukan terhadap bilangan biner dari bilangan bulat tersebut.
- Contoh :

```
int a = 100, b = 50, hasil2;
bool c = false, d = true, hasil1;
hasil1 = (a <= b) && (c ^ !d);
```
→ Karena ekspresi **(a <= b)** telah menghasilkan nilai **False**, maka akan langsung dikembalikan nilai **False** ke variabel **hasil1** tanpa dievaluasi lagi ekspresi **(c ^ !d)**.

```
hasil1 = (a <= b) & (c ^ !d);
```
→ Meskipun ekspresi **(a <= b)** telah menghasilkan nilai **False**, ekspresi **(c ^ !d)** akan tetap dievaluasi dan menghasilkan nilai **True**, barulah dikembalikan nilai **False** ke variabel **hasil1**.

```
hasil2 = a & b;
```
→ Nilai variabel **a** = 100 (dalam biner = "**0110 0100**") dilakukan logika **AND** dengan nilai variabel **b** = 50 (dalam biner = "**0011 0010**"), dan dikembalikan hasil biner = "**0010 0000**" (dalam desimal = **32**) ke variabel **hasil2**.

```
hasil1 = (c ^ !d) || (a != b);
```
→ Karena ekspresi **(c ^ !d)** telah menghasilkan nilai **True**, maka akan langsung dikembalikan nilai **True** ke variabel **hasil1** tanpa dievaluasi lagi ekspresi **(a != b)**.

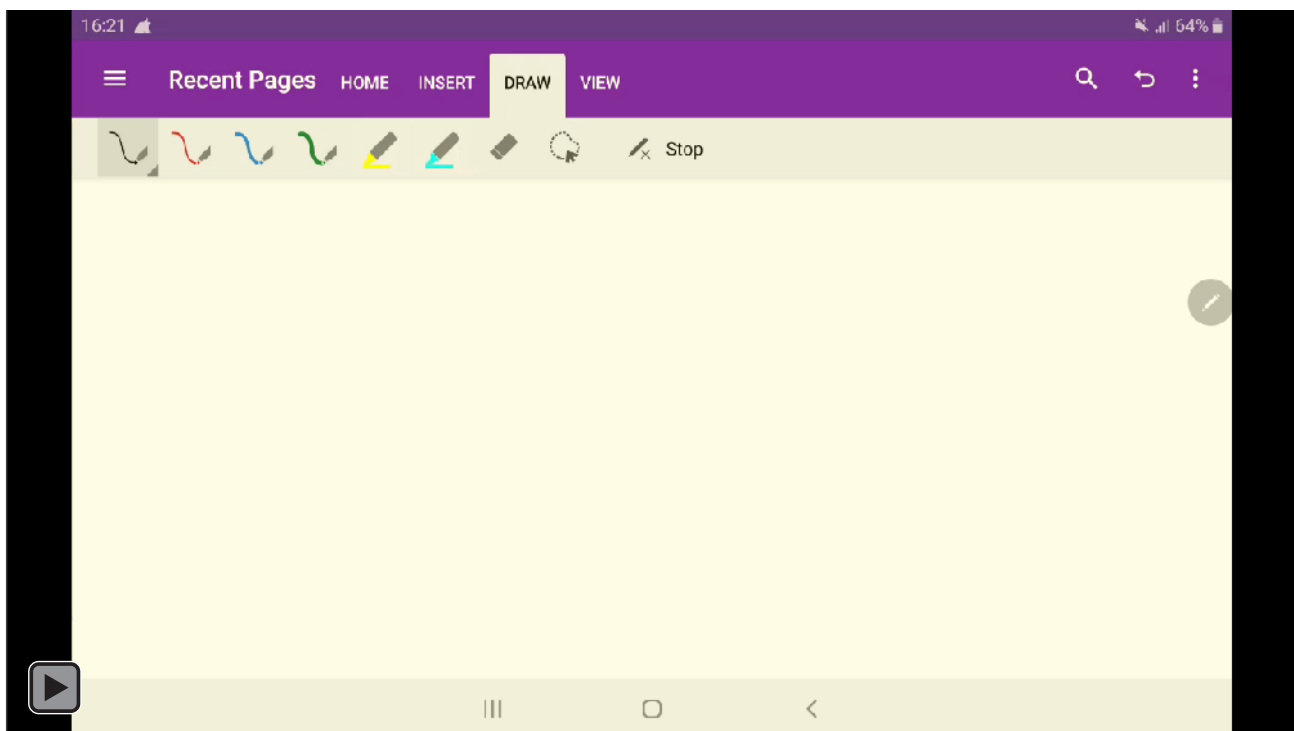
PENJELASAN DARI SLIDE ke-9 & 10

`hasil1 = (c ^ !d) | (a != b);` → Meskipun ekspresi **(c ^ !d)** telah menghasilkan nilai True, ekspresi **(a != b)** akan tetap dievaluasi dan menghasilkan nilai **True**, barulah dikembalikan nilai **True** ke variabel **hasil1**.

`hasil2 = a | b;` → Nilai variabel **a** = 100 (dalam biner = "**0110 0100**") dilakukan logika **OR** dengan nilai variabel **b** = 50 (dalam biner = "**0011 0010**"), dan dikembalikan hasil biner = "**0111 0110**" (dalam desimal = **118**) ke variabel **hasil2**.

[Slide 10]
Dari contoh kode program tersebut, dapat dijelaskan :

- "`var1 = (var2 >= 0) && (var2 < 100);`"
  Karena ekspresi **(var2 >= 0)** menghasilkan nilai **True**, maka akan dilanjutkan evaluasi ekspresi **(var2 < 100)** dan menghasilkan nilai **True**, serta di simpan ke variabel **var1**.
- "`var1 = (var3 <= 0) || (var3 >= 100);`"
  Karena ekspresi **(var3 <= 0)** telah menghasilkan **True**, maka akan langsung dikembalikan nilai **True** ke variabel **var1** tanpa dievaluasi lagi ekspresi **(var3 >= 100)**.

[Video Ilustrasi]
Note : Memerlukan Adobe Flash Player & Video memiliki Audio.

PENJELASAN DARI SLIDE ke-11 & 12

[Slide 11]

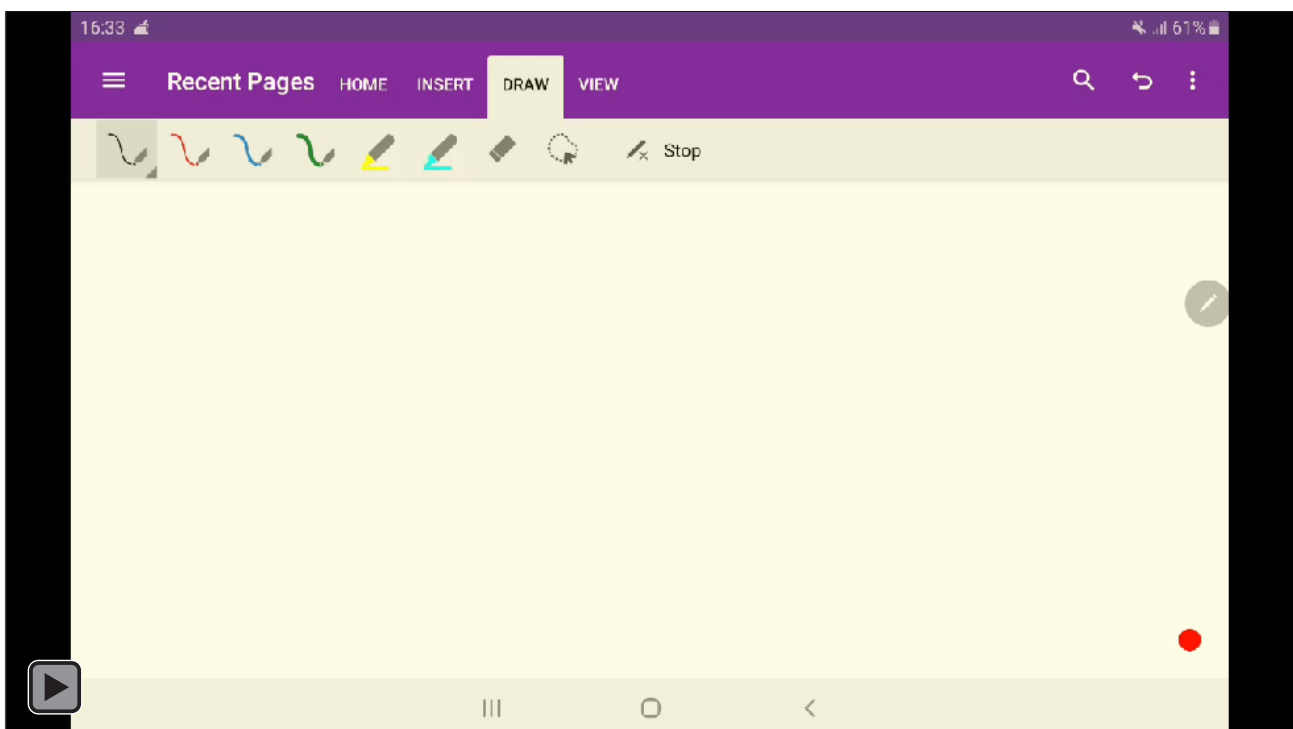
- Merupakan daftar operator bitwise pada Visual C#, beserta alternatif berbeda.
- "`hasil = 214 >> 2;`" → Akan bernilai **53**, dimana dihasilkan dari nilai **214** dalam biner "**1101 0110**" dilakukan **SHIFT RIGHT** (pergeseran ke kanan) sebesar **2 bit**, sehingga biner "**1101 0110→**" digeser **2 bit** ke kanan menjadi "**0011 0101**" (menjadi desimal **53**).
- "`hasil = 93 << 4;`" → Akan bernilai **1488**, dimana dihasilkan dari nilai **93** dalam biner "**0101 1101**" dilakukan **SHIFT LEFT** (pergeseran ke kiri) sebesar **4 bit**, sehingga biner "**←0101 1101**" digeser **4 bit** ke kiri menjadi "**0101 1101 0000**" (menjadi desimal **1488**).

[Slide 12]

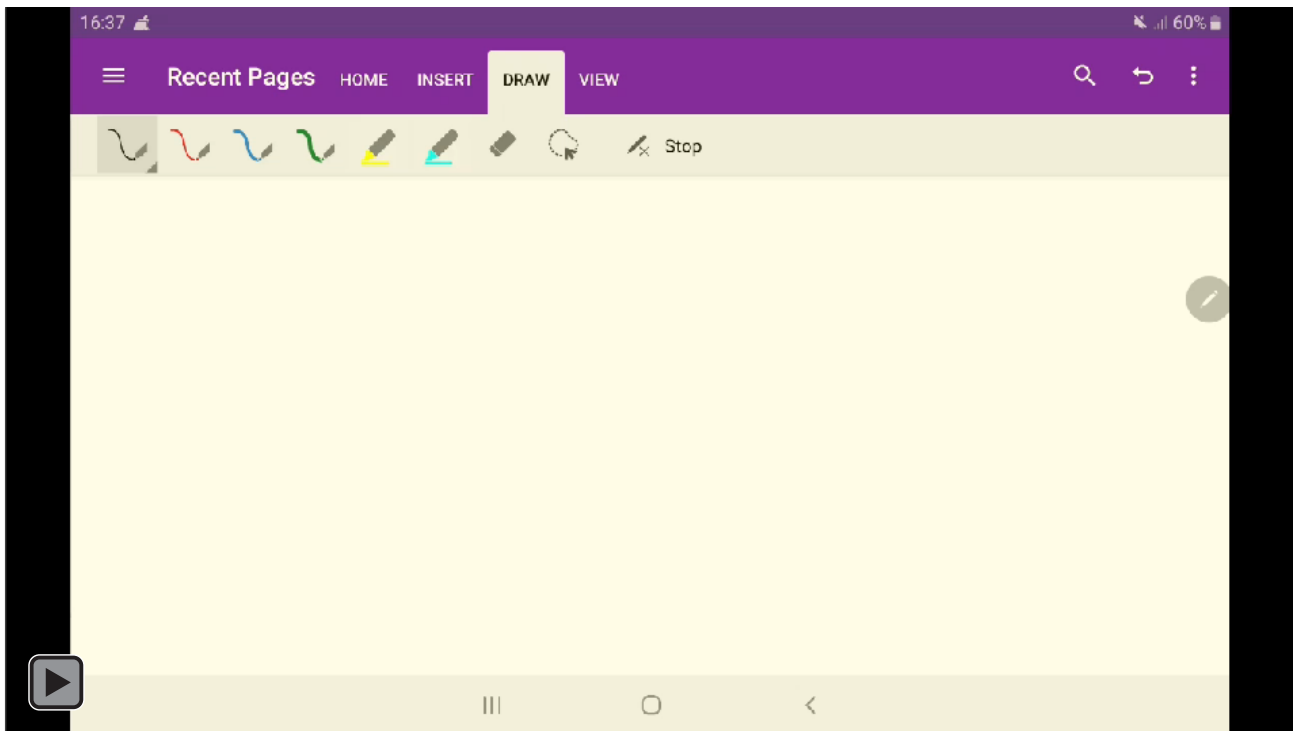
Dari contoh kode program tersebut, dapat dijelaskan :

- "`var1 = var2 >> 1;`" → Nilai variabel **var2** (bernilai **172**) dalam biner "**1010 1100**" dilakukan **SHIFT RIGHT** (pergeseran ke kanan) sebesar **1 bit**, sehingga biner "**1010 1100→**" digeser **1 bit** ke kanan menjadi "**0101 0110**" (menjadi desimal **86**).
- "`var1 = var3 << 2;`" → Nilai variabel **var3** (bernilai **149**) dalam biner "**1001 0101**" dilakukan **SHIFT LEFT** (pergeseran ke kiri) sebesar **2 bit**, sehingga biner "**←1001 0101**" digeser **2 bit** ke kiri menjadi "**0010 0101 0100**" (menjadi desimal **596**).
- "`var3 >>= 3;`" → identik dengan "`var3 = var3 >> 3;`"

[Video Ilustrasi]

Note : Memerlukan Adobe Flash Player & Video memiliki Audio.

PENJELASAN DARI SLIDE ke-13

Operator ternary sama dengan fungsi IF pada Microsoft Excel (MK Otomasi Perkantoran di Semester 1).

[Video Ilustrasi]
Note : Memerlukan Adobe Flash Player & Video memiliki Audio.

PENJELASAN DARI SLIDE ke-14 s/d 19

[Slide 14 & 15]

- Contoh percabangan IF dengan 1 kondisi :

```
if (nilai > 50)
    Console.Write("LULUS");
```

→ Jika variabel **nilai** > 50, maka akan dicetak output **LULUS**, jika tidak, maka tidak terjadi apa-apa.

- Contoh percabangan IF dengan 2 kondisi :

```
if (nilai > 50)
    Console.Write("LULUS");
else
    Console.Write("GAGAL");
```

→ Jika variabel **nilai** > 50, maka akan dicetak output **LULUS**.
→ Jika variabel **nilai** <= 50, maka akan dicetak output **GAGAL**.

- Contoh percabangan IF dengan banyak kondisi :

```
if (IPK >= 3.50f)
    Console.Write("MANTAP");
else if (IPK >= 3.00f)
    Console.Write("BAGUS");
else if (IPK >= 2.75f)
    Console.Write("LUMAYAN");
else
    Console.Write("CUKUP");
```

→ Jika variabel **IPK** >= 3.50, maka akan dicetak output **MANTAP**.
→ Jika variabel **IPK** >= 3.00, maka akan dicetak output **BAGUS**.
→ Jika variabel **IPK** >= 2,75, maka akan dicetak output **LUMAYAN**.
→ Jika variabel **IPK** < 2,75, maka akan dicetak output **CUKUP**.

[Slide 16]

- Penggunaan blok **{ }** di dalam percabangan IF wajib digunakan apabila jumlah baris statement kode program lebih dari 1 (satu) baris.
- Apabila jumlah baris statement kode program di dalam percabangan IF hanya 1 (satu) baris saja, maka penggunaan blok **{ }** bersifat opsional, artinya digunakan atau tidak digunakan, tidak akan terjadi Error.

[Slide 17]

Struktur percabangan IF ini digunakan apabila di dalam suatu perulangan IF terdapat lagi perulangan IF.

[Slide 18 & 19]

Dari contoh kode program tersebut, dapat dijelaskan :

- Jika "`if (var1 == var2)`", maka akan diset warna tulisan menjadi hijau, dan kemudian dicetak (output) nilai dari kedua variabel tersebut (**var1** & **var2**).
- Dikarenakan di dalam percabangan "`if (var1 == var2)`" terdapat 2 (dua) baris statement kode program (mengubah warna tulisan & cetak output), maka digunakan blok **{ }** untuk kedua baris statement kode program tersebut.
- Jika "`else if (var1 < var2)`", maka akan diset warna tulisan menjadi biru, dan kemudian dicetak (output) nilai dari kedua variabel tersebut (**var1** & **var2**).
- Jika tidak terjadi kondisi "`if (var1 == var2)`" dan "`else if (var1 < var2)`", maka akan dijalankan kondisi "`else`", yakni akan diset warna tulisan menjadi kuning, dan kemudian dicetak (output) nilai dari kedua variabel tersebut (**var1** & **var2**).
- Dikarenakan di dalam percabangan "`else if (var1 < var2)`" dan "`else`" terdapat 2 (dua) baris statement kode program (mengubah warna tulisan & cetak output), maka digunakan blok **{ }** untuk kedua baris statement kode program tersebut.

PENJELASAN DARI SLIDE ke-20 s/d 27

[Slide 20]
Merupakan opsi lain dari struktur percabangan, akan tetapi percabangan SWITCH ini hanya berlaku untuk membandingkan kondisi nilai yang sudah tetap/fix (tidak berlaku untuk nilai jangkauan/range).

[Slide 21 & 22]
- **<testVar>** merupakan variabel yang akan di cek kondisinya dengan percabangan SWITCH.
- **<comparisonVal1>**, **<comparisonVal1>**, **<comparisonVal1>** merupakan nilai dari variabel **<textVar>** yang akan di cek.
- Setiap blok kondisi, wajib diakhiri dengan statement "**break;**".
- Wajib menggunakan blok **{ }** sesuai dengan strukturnya.
- Fungsi dari statement "**default:**" identik dengan "**else**" pada percabangan IF, yakni blok kode program apabila semua kondisi di atas tidak ada yang memenuhi.
- Contoh :

```
switch (rambu)
{
    case "MERAH":
        Console.Write("BERHENTI");
```
→ Jika nilai variabel **rambu** = "**MERAH**", maka akan dicetak output **BERHENTI**.
```
        break;
    case "KUNING":
        Console.Write("SIAP-SIAP");
```
→ Jika nilai variabel **rambu** = "**KUNING**", maka akan dicetak output **SIAP-SIAP**.
```
        break;
    case "HIJAU":
        Console.Write("JALAN");
```
→ Jika nilai variabel **rambu** = "**HIJAU**", maka akan dicetak output **JALAN**.
```
        break;
    default:
        Console.Write("ERROR");
```
→ Jika nilai variabel **rambu** bernilai selain "**MERAH**", "**KUNING**", dan "**HIJAU**", maka akan dicetak output **ERROR**.
```
        break;
}
```

[Slide 23]
- "**goto case <comparisonVal>;**" digunakan untuk lintas dari 1 kondisi ke kondisi lain selama dalam percabangan SWITCH yang sama.
- Contoh :

```
switch (rambu)
{
    case "MERAH":
        Console.Write("BERHENTI");
```
→ Jika nilai variabel **rambu** = "**MERAH**", maka akan dicetak output **BERHENTI**.
```
        break;
```

PENJELASAN DARI SLIDE ke-20 s/d 27

```
        case "KUNING":
            Console.Write("SIAP-SIAP");
            goto case "HIJAU";
            break;
        case "HIJAU":
            Console.Write("JALAN");
            break;
        default:
            Console.Write("ERROR");
            break;
    }
```

- `Console.Write("SIAP-SIAP");` → Jika nilai variabel **rambu** = "**KUNING**", maka akan dicetak output **SIAP-SIAP**.
- `goto case "HIJAU";` → Loncat ke case "**HIJAU**", sehingga akan lanjut dicetak output **JALAN**.
- `Console.Write("JALAN");` → Jika nilai variabel **rambu** = "**HIJAU**", maka akan dicetak output **JALAN**.
- `Console.Write("ERROR");` → Jika nilai variabel **rambu** bernilai selain "**MERAH**", "**KUNING**", dan "**HIJAU**", maka akan dicetak output **ERROR**.

[Slide 24]

- Untuk membandingkan nilai yang lebih dari 1 (satu), tetapi bukan berupa nilai jangkauan/range.
- Contoh :

```
    switch (rambu)
    {
        case "MERAH":
        case "KUNING":
        case "HIJAU":
            Console.Write("RAMBU BERFUNGSI");
            break;
        default:
            Console.Write("ERROR");
            break;
    }
```

- `Console.Write("RAMBU BERFUNGSI");` → Jika nilai variabel **rambu** = "**MERAH**", "**KUNING**", atau "**HIJAU**", maka akan dicetak output **RAMBU BERFUNGSI**.
- `Console.Write("ERROR");` → Jika nilai variabel **rambu** bernilai selain "**MERAH**", "**KUNING**", dan "**HIJAU**", maka akan dicetak output **ERROR**.

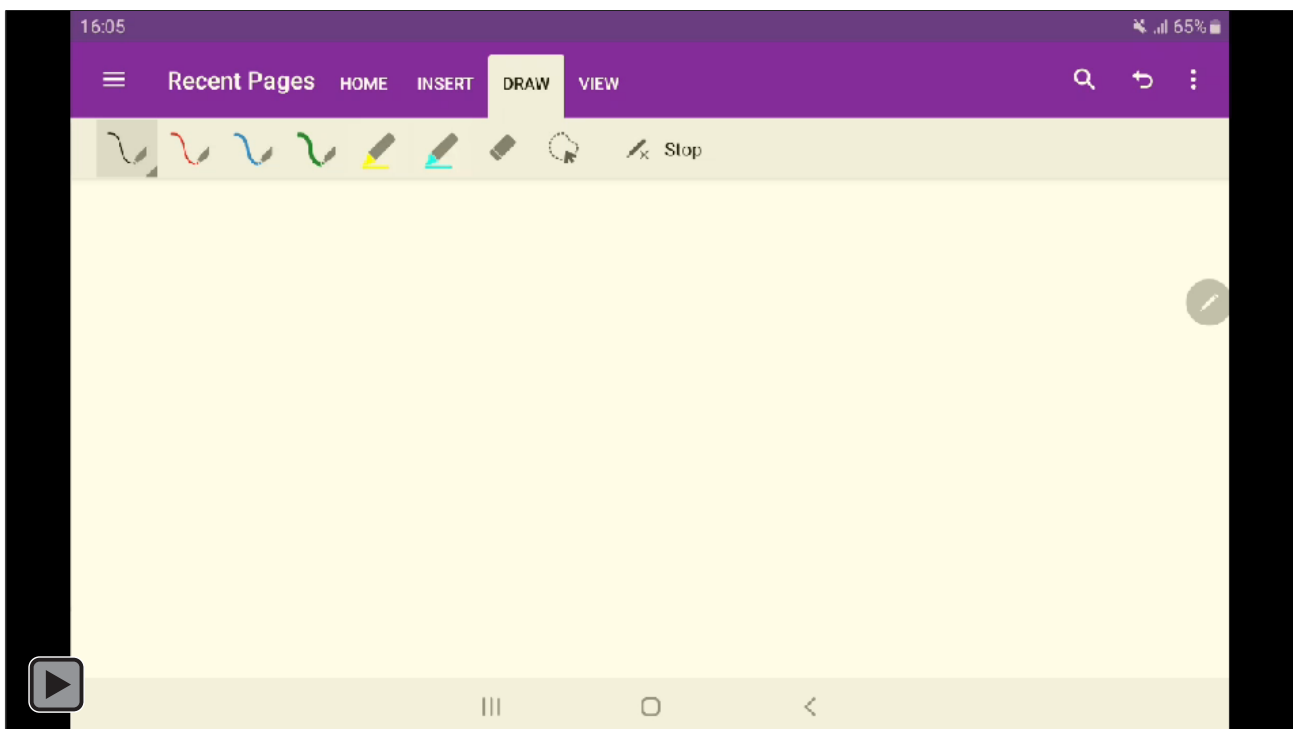
[Slide 25 s/d 27]

Dari contoh kode program tersebut, dapat dijelaskan :

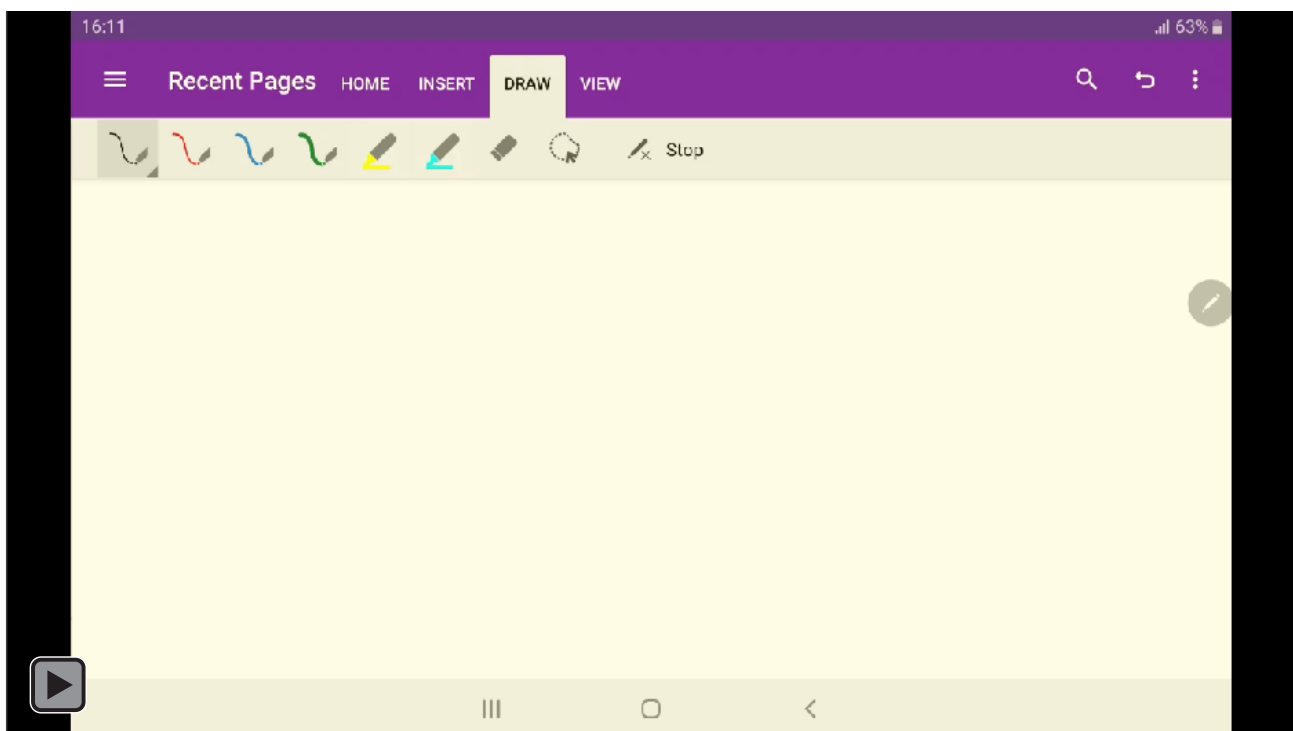
- Jika nilai variabel **var** = **0**, maka akan dicetak output Nol.
- Jika nilai variabel **var** = **1**, **3**, **5**, **7**, atau **9**, maka akan dicetak output **Ganjil**.
- Jika nilai variabel **var** = **2**, **4**, **6**, atau **8**, maka akan dicetak output **Genap**.
- Jika nilai variabel **var** selain **0 s/d 9** (Nol, Ganjil, dan Genap), maka akan dicetak output **Negatif atau > 9**.

PENJELASAN DARI SLIDE ke-28

- Perulangan **DO** akan terus dilakukan selama kondisi **<Test>** bernilai **True**.
- Jumlah perulangan yang terjadi pada perulangan DO tergantung pada kapan hasil kondisi **<Test>** bernilai **False**.
- Perulangan DO minimal akan dijalankan 1 (satu) kali perulangan, karena pengecekan kondisi **<Test>** dilakukan di akhir blok perulangan **{ }**.
- Dari contoh kode program tersebut, terlihat bahwa perulangan DO akan dilakukan terus selama nilai variabel **i <= 10**.

[Video Ilustrasi]
Note : Memerlukan Adobe Flash Player & Video memiliki Audio.

PENJELASAN DARI SLIDE ke-29

- Perulangan **WHILE** akan terus dilakukan selama kondisi **<Test>** bernilai **True**.
- Jumlah perulangan yang terjadi pada perulangan WHILE tergantung pada kapan hasil kondisi **<Test>** bernilai **False**.
- Pengecekan kondisi **<Test>** pada perulangan WHILE dilakukan sebelum perulangan dilakukan, sehingga dimungkinkan tidak ada terjadi perulangan sama sekali.
- Dari contoh kode program tersebut, terlihat bahwa perulangan WHILE akan dilakukan terus selama nilai variabel **i <= 10**.

[Video Ilustrasi]
Note : Memerlukan Adobe Flash Player & Video memiliki Audio.

PENJELASAN DARI SLIDE ke-30 s/d 32

[Slide 30]

- Perulangan **FOR** memiliki 3 parameter, yakni **<initialization>**, **<condition>**, dan **<operation>**.
- Pada perulangan FOR, jumlah perulangan yang terjadi dapat ditentukan berdasarkan parameter **<initialization>**, **<condition>**, dan **<operation>**.
- **<initialization>** berupa kondisi awal, dapat berupa inisialisasi nilai awal dari suatu variabel.
- **<condition>** berupa kondisi akhir, dapat berupa nilai dari suatu variabel. Jika nilai variabel dari inisialisasi **<initialization>** sudah mencapai nilai yang ditentukan **<condition>**, maka perulangan selesai.
- **<operation>** berupa suatu operasi yang diset terhadap nilai variabel inisialisasi agar dapat mencapai kondisi **<condition>**.

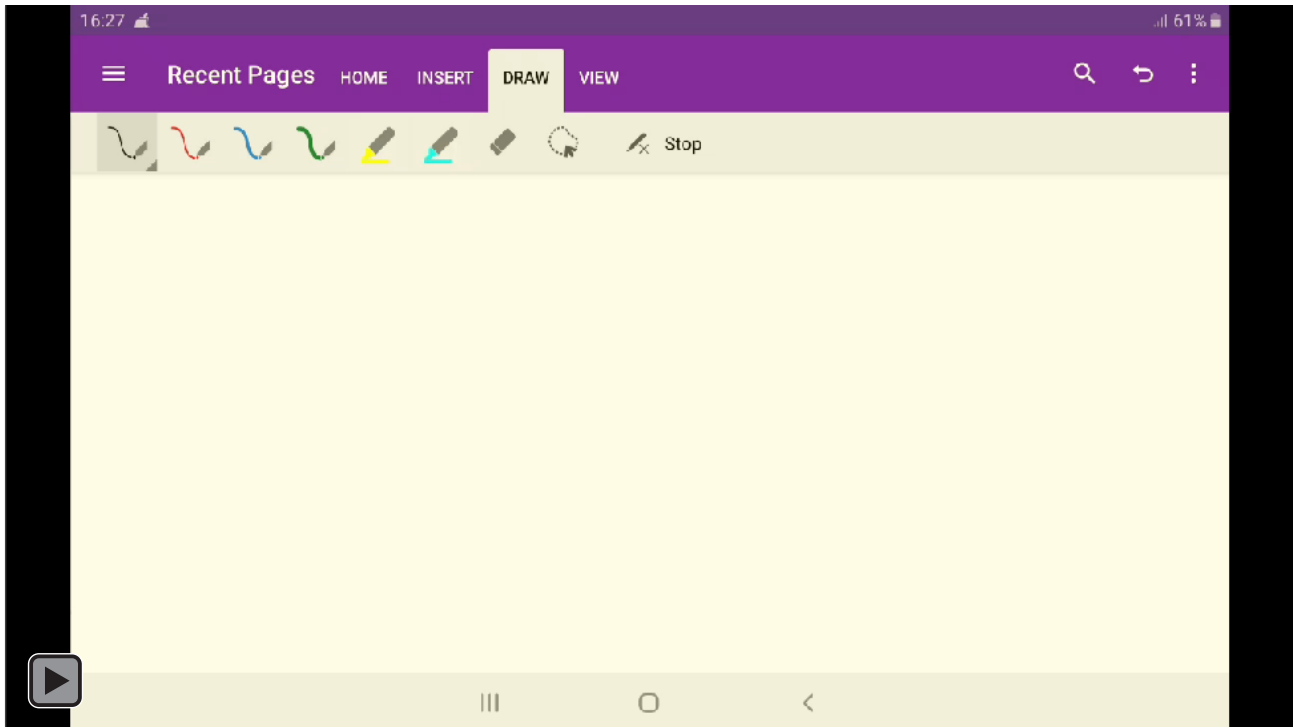
[Slide 31]

- Dari contoh kode program pertama tersebut, dapat dijelaskan :
  - **<initialization>** berupa "**i = 1**", dimana dinyatakan nilai awal variabel **i = 1**.
  - **<condition>** berupa "**i <= 5**", dimana perulangan akan terjadi terus selama nilai variabel **i <= 5**.
  - **<operation>** berupa "**i++**", dimana nilai variabel mulai dari nilai 1 (initialization) hingga mencapai 5 (condition), akan dilakukan operasi **i++**.
  - Jumlah perulangan yang terjadi adalah sebanyak 5 (lima) kali, yakni di saat nilai variabel **i** = 1, 2, 3, 4, dan 5.
- Dari contoh kode program kedua tersebut, dapat dijelaskan :
  - **<initialization>** berupa "**i = 5**", dimana dinyatakan nilai awal variabel **i = 5**.
  - **<condition>** berupa "**i > 0**", dimana perulangan akan terjadi terus selama nilai variabel **i > 0**.
  - **<operation>** berupa "**i--**", dimana nilai variabel mulai dari nilai 5 (initialization) hingga mencapai 1 (condition), akan dilakukan operasi **i--**.
  - Jumlah perulangan yang terjadi adalah sebanyak 5 (lima) kali, yakni di saat nilai variabel **i** = 5, 4, 3, 2, dan 1.

[Slide 32]

- Dari contoh kode program pertama tersebut, dapat dijelaskan :
  - **<initialization>** berupa "**i = 1**", dimana dinyatakan nilai awal variabel **i = 1**.
  - **<condition>** berupa "**i <= 10**", dimana perulangan akan terjadi terus selama nilai variabel **i <= 10**.
  - **<operation>** berupa "**i += 2**", dimana nilai variabel mulai dari nilai 1 (initialization) hingga mencapai 10 (condition), akan dilakukan operasi **i += 2** (increment sebesar 2).
  - Jumlah perulangan yang terjadi adalah sebanyak 5 (lima) kali, yakni di saat nilai variabel **i** = 1, 3, 5, 7, dan 9.
- Dari contoh kode program kedua tersebut, dapat dijelaskan :
  - **<initialization>** berupa "**i = 10**", dimana dinyatakan nilai awal variabel **i = 10**.
  - **<condition>** berupa "**i > 0**", dimana perulangan akan terjadi terus selama nilai variabel **i > 0**.
  - **<operation>** berupa "**i -= 2**", dimana nilai variabel mulai dari nilai 10 (initialization) hingga mencapai 1 (condition), akan dilakukan operasi **i -= 2** (decrement sebesar 2).
  - Jumlah perulangan yang terjadi adalah sebanyak 5 (lima) kali, yakni di saat nilai variabel **i** = 10, 8, 6, 4, dan 2.

PENJELASAN DARI SLIDE ke-30 s/d 32

[Video Ilustrasi]
Note : Memerlukan Adobe Flash Player & Video memiliki Audio.

PENJELASAN DARI SLIDE ke-33 s/d 36

[Slide 33]
Perulangan yang sedang terjadi, baik perulangan **DO**, **WHILE**, atau **FOR**, dapat dilakukan interupsi perulangan, dimana perulangan yang sedang terjadi dan belum memenuhi kondisi perulangan selesai, dapat dihentikan secara mendadak.

[Slide 34]
Dari contoh kode program tersebut, dapat dijelaskan :

- Perulangan **WHILE** akan dijalankan selama nilai variabel **i <= 10**.
- Perulangan **WHILE** akan dihentikan di saat nilai variabel **i** bernilai 6 (belum mencapai 10).

[Slide 35]
Dari contoh kode program tersebut, dapat dijelaskan :

- Perulangan **FOR** akan dilakukan mulai dari nilai variabel **i = 1** (initialization) hingga mencapai **10** (condition), dengan operasi **i++**.
- Jumlah perulangan yang harusnya terjadi adalah sebanyak 10 (sepuluh) kali, yakni di saat nilai variabel **i = 1 s/d 10**.
- Akan tetapi, perulangan yang terjadi hanya 5 (lima) kali, yakni di saat variabel **i** = 1, 3, 5, 7, dan 9.
- Hal ini terjadi karena di saat variabel **i** = 2, 4, 6, 8, dan 10, maka perulangan yang sedang terjadi akan diabaikan, dan dilanjutkan ke perulangan berikutnya.

[Slide 36]
Dari contoh kode program tersebut, dapat dijelaskan :

- Terdapat 1 label yakni **exitPoint**.
- Perulangan **WHILE** akan dijalankan selama nilai variabel **i <= 10**.
- Perulangan **WHILE** akan dihentikan di saat nilai variabel **i** bernilai 6 (belum mencapai 10), dan kode program akan langsung pindah ke kode program pada label **exitPoint**.
- Baris kode "`Console.WriteLine("Bagian ini tidak akan pernah dicapai");`" tidak akan pernah dieksekusi oleh program.